Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 7104764, 14 pages
https://doi.org/10.1155/2018/7104764

Research Article

Shared Variable Extraction and Hardware Implementation for Nonlinear Boolean Functions Based on Swarm Intelligence

Longmei Nan,[1,2] **Xiaoyang Zeng**,[1] **Yiran Du**,[2] **Zibin Dai**,[2] **and Lin Chen**[2]

[1]*ASIC & System State Key Laboratory of Fudan University, Shanghai 201203, China*
[2]*Institute of Information Science and Technology, Zhengzhou 450001, China*

Correspondence should be addressed to Longmei Nan; lnan13@fudan.edu.cn

Received 25 December 2017; Accepted 30 July 2018; Published 9 August 2018

Academic Editor: Ricardo Soto

To solve the problem of complex relationships among variables and the difficulty of extracting shared variables from nonlinear Boolean functions (NLBFs), an association logic model of variables is established using the classical Apriori rule mining algorithm and the association analysis launched during shared variable extraction (SVE). This work transforms the SVE problem into a traveling salesman problem (TSP) and proposes an SVE based on particle swarm optimization (SVE-PSO) method that combines the association rule mining method with swarm intelligence to improve the efficiency of SVE. Then, according to the shared variables extracted from various NLBFs, the distribution of the shared variables is created, and two corresponding hardware circuits, Element A and Element B, based on cascade lookup table (LUT) structures are proposed to process the various NLBFs. Experimental results show that the performance of SVE via SVE-PSO method is significantly more efficient than the classical association rule mining algorithms. The ratio of the rules is 80.41%, but the operation time is only 21.47% when compared to the Apriori method, which uses 200 iterations. In addition, the area utilizations of Element A and Element B expended by the NLBFs via different parallelisms are measured and compared with other methods. The results show that the integrative performances of Element A and Element B are significantly better than those of other methods. The proposed SVE-PSO method and two cascade LUT-structure circuits can be widely used in coarse-grained reconfigurable cryptogrammic processors, or in application-specific instruction-set cryptogrammic processors, to advance the performance of NLBF processing and mapping.

1. Introduction

1.1. Background. Cryptographic algorithms play a critical role in information security, which has broad applications. Currently, nonlinear Boolean functions (NLBFs) are being used in cryptographic algorithms widely. The performance of NLBFs processing and mapping influences the performance of cryptographic algorithms tremendously, it is indicated that the characteristics of the shared variables extracted from NLBFs must be determined so the efficient hardware circuits and mapping of NLBFs can be implemented.

The distribution of NLBFs with shared variables is difficult to create because the variables possess complicated expression types with unique design principles. The distribution also affects or even determines the organization of the lookup table- (LUT-) based logic elements, further impacting the hardware performance (in terms of area, speed, power, and area utilization) of the whole computing architecture. Although many cascading LUT-based structures for Boolean functions have been proposed, the methods have usually been based on experimental results or engineering experience, which lack a standard and basic conducting theory.

Shared input ports are common in LUT-based structures because they complete more powerful functionalities and achieve a more advanced performance. Adaptive logic modules (ALMs) developed by Mike Hutton [1] can realize multiple Boolean functions and achieve a 15% performance increase and a 12% decrease in the area versus standard basic logical element (BLE) in field-programmable gate arrays (FPGAs). Jason H. Anderson [2] projected function generators that decompose a k-variable function into two (k-1)-variable expression using the Shannon decomposition theorem. Next, a diverse LUT-based structure

was devised according to the decomposition results. Other existing research studies [3–5] have also formed Boolean functions by applying LUT-based structures with shared inputs.

In addition, there is abundant research on the features and extraction methods of shared variables in general computing. Earlier studies were concerned with the relationships between the LUT size (input number of LUT, K), cluster size (number of LUTs per cluster, N), and cluster inputs (input number of a cluster, I) by testing all types of LUT-based structures. The logic utilization of cluster-based modules was analyzed by Jonathan Rose [6] and considered the effect of three parameters: area, delay, and the area-delay product. This group experimentally determined the relationship between K and N by achieving a peak utilization ratio of 98%. Furthermore, they revisited the issue and determined the relationship between parameters via (1) across all architectures with LUT sizes that range between 2 and 7 input ports and with a cluster size from 1 to 10 LUTs [7].

$$I = \frac{K}{2} \times (N + 1) \tag{1}$$

To reduce I to K, multiply it by N shared inputs (the shared variables of the Boolean expressions); if $N \neq 1$ is also proved in [8–10], then the number of shared inputs is

$$I_{shared} = K \times N - \frac{K}{2} \times (N + 1) = \frac{K}{2} \times (N - 1) \tag{2}$$

The above considerations are directed at general computing, which may provide some inspiration to achieve the distribution for shared variables. There are a few important characteristics of NLBFs. Research initiatives have not considered solving the problem from the expressions themselves, and there is, indeed, no efficient method for obtaining the shared variables of Boolean expressions.

In recent studies, researchers have advanced investigations into variables, expressions, and other related factors. A method [11] for *and-term*s and variables that is statistical in nature was proposed to realize the critical arithmetic unit of NLBFs. However, the results usually depend on a subjective verdict causing unreasonable settings for the shared inputs, which also achieves low efficiency and has limited applicability when there are many terms in the complex relationship between terms, which are ubiquitous in the various design principles of NLBFs. A reconfigurable structure was provided in [12], according to the results. This indicates that an achieved throughput of 7.6 Gbps in a 32-way parallel architecture acquired high utilization but also failed to have a profound deliberation of the shared variables.

Although the parameters (*and-term*s, variables, and others) of NLBFs have been analyzed, the most important elements, the shared variables, have not been studied profoundly. The essence of shared variable extraction is the knowledge of the incidence and relationship of the items. Association rule mining is a significant approach when finding the relationship between items; this approach is also an undirected mining method that discovers the valuable association relationships found in large amounts of data. To obtain the association logic of the variables, the Boolean expressions of NLBFs in cryptographic algorithms are analyzed. In addition, based on the common characteristics between the frequent items and shared variables (the more frequent an item (variable) is, the more probability it has of being a frequent item set (shared variable)), association rule mining is applied to the extraction of shared variables to achieve more reasonable shared inputs.

However, there are many variables and complex expressions in various NLBFs. The classical association rule mining algorithm, Apriori or frequent pattern (FP) growth, provides a low efficiency and bears a heavy computing load. To overcome these disadvantages, this study proposes an efficient extraction algorithm for shared variables that incorporates swarm intelligence while also designing the corresponding hardware elements based on the distribution of the extracted shared variables.

1.2. Motivation and Contribution. Having studied the principle and process of association rule mining, this work suggests that the association rule analysis can be perfectly applied to the extraction of shared variables from NLBFs. Meanwhile, LUT-based logic elements are constructed according to the distribution of shared variables. In this study, the shared variable extraction algorithm for NLBFs in cryptographic algorithms is investigated. The main contributions of our work are listed as follows:

(1) Research on association rule mining is conducted and a logic model based on the association rules is established. This model combines the generation of frequent items with the extraction of shared variables.

(2) To improve the efficiency of the extraction algorithm, this study transforms the shared variable extraction (SVE) into the traveling salesman problem (TSP). Based on the similarity between the shortest path and the maximum *support*, particle swarm optimization (PSO) is applied to the undirected path treating the variables as points.

(3) The distribution of shared variables is achieved in various NLBFs. Based on the distribution of the shared variables, two cascade LUT-structures Element A and Element B are projected to satisfy the computation of NLBFs.

These novel expressions provide a perfect extraction for shared variables and two novel hardware elements for NLBFs. The extraction algorithm devised in our work will be a helpful reference for researchers dealing with LUT-based structures, especially with the hardware structures of NLBFs in cryptographic algorithms. This paper is arranged as follows. Section 2 establishes the association logic for NLBFs by applying the Apriori algorithm. The PSO method is incorporated to improve efficiency. In Section 3, SVE based on PSO (SVE-PSO) is proposed, and the procedure is determined. In Section 4, the hardware elements, based on the distribution of shared variables, are proposed to process NLBFs. The performances of SVE-PSO and the hardware elements based on shared variables are evaluated in Section 5. We conclude the paper in Section 6.

2. Association Logic of NLBFs and PSO

Due to different principles and diverse security requirements, various NLBFs possess different features. To create a distribution of shared variables, the association logic should be obtained first.

2.1. Association Logic Analysis of NLBFs

Definition 1. A Boolean function is a function with the domain of $\{0,1\}^n$ and a range of $\{0,1\}$. It can be expressed in regular type as (AND-XOR):

$$f = c_0 + \sum_i c_{a_i} x_i + \sum_{i>j} c_{a_i a_j} x_i x_j + \cdots + c_{a_{n-1}\cdots a_1 a_0} x_{n-1} \cdots x_1 x_0 \quad (3)$$

Each x_i represents the Boolean variable, respectively, $x_i x_j$ called *and-term* represents the operation of x_i "AND" x_j, and + represents the operation of "XOR". The value of each coefficient c_{a_i} is "1" or "0", which indicates whether the corresponding term is included in the expression. Each *and-term* consists of "AND" and the variable number is its corresponding order. The maximum order of all *and-term*s is the order of the Boolean function, and the Boolean function is called NLBF when its order is greater than 1.

Definition 2. Association item refers to the combination of the Boolean variables that appear at the same time, that is, the combination of the common variables. The detailed definition is as follows. There are m samples $I = \{i_1, i_2, \ldots, i_m\}$ in item sets I. D is the task database related to all tasks and T is a subset of the data items; that is, $T \subset I$. The goal of association mining is to find the correlation of task A and task B noted as $A \longrightarrow B$ ($A \in I$, $B \in I$, and $A \cap B = \emptyset$). A is called the *association antecedent* item and B is called the *association succedent* item.

Definition 3. Support refers to the frequency of a Boolean variable or certain Boolean variables combined in all Boolean terms, and it mirrors the frequency and applicability of the association rules. The greater the *support*, the stronger the association of the Boolean terms, and therefore the greater the possibility of the implementation using the same LUT. *Support*(A) is the ratio of task A and is equal to its probability. *Support*(A, B) is the joint probability of A and B, meaning the item includes A and B simultaneously. In this paper, *Support* is marked as L.

Definition 4. Confidence refers to the ratio of item B when item A is included in the data set. It represents the credibility and accuracy of the related items. The *confidence* (A⟶B) is equal to the conditional probability $p(B \mid A)$.

Association rule mining can acquire the relationships among the items in a dataset via mathematical logic. Apriori [13] is the classical association rule mining algorithm. This algorithm searches for the frequent item sets iteratively; there are two steps to accomplish the process:

Table 1: *And-term* sets of NLBF.

And-term event (item)	Variable item sets
Term 1	$\{\{x_2\}, \{x_4\}, \{x_5\}, \{x_6\}\}$
Term 2	$\{\{x_1\}, \{x_5\}, \{x_6\}\}$
Term 3	$\{\{x_2\}, \{x_4\}\}$
Term 4	$\{\{x_3\}, \{x_4\}\}$
Term 5	$\{\{x_2\}, \{x_5\}, \{x_6\}\}$
Term 6	$\{\{x_1\}, \{x_2\}, \{x_4\}\}$

Table 2: *Support* calculation for items L1.

Variable item	Support count
$\{x_1\}$	2
$\{x_2\}$	4
$\{x_3\}$	1
$\{x_4\}$	4
$\{x_5\}$	3
$\{x_6\}$	3

Table 3: *Support* count for variable sets C1.

Variable item	*Support* count
$\{x_1\}$	3
$\{x_2\}$	3
$\{x_4\}$	4
$\{x_5\}$	3
$\{x_6\}$	3

(1) Achieve the frequent item sets. To do this, find the frequent item sets whose *support*s are larger than *min_sup*.

(a) Set *min_sup* (minimum *support*).

(b) Scan the database D and determine the frequent 1-item sets that contain only one variable.

(c) Circle the search process until all frequent item sets are found.

(2) Generate the strong association rules.

(a) Set *min_conf* (minimum *confidence*).

(b) Count the *confidence*s and consider all frequent item sets whose *confidence*s are greater than *min_conf* as the strong association rules.

According to this procedure, the extraction process for shared variables is introduced with an example of the following NLBF, whose expression is $f(x_1, x_2, x_3, x_4, x_5, x_6) = x_2x_4x_5x_6 + x_1x_5x_6 + x_2x_4 + x_3x_4 + x_2x_5x_6 + x_1x_2x_4$, whose six *and-term*s are $x_2x_4x_5x_6, x_1x_5x_6, x_2x_4, x_3x_4, x_2x_5x_6, x_1x_2x_4$ marked as Term1~Term6 correspondingly. Item sets in this NLBF expression are shown in Table 1. All *and-term*s have their own identification (ID) and there are 6 variables $x_1 \sim x_6$ in this NLBF.

(a) Scan the *and-term* sets completely and count their *support*s L1 to find the initial item sets shown in Table 2.

(b) Set the *min_sup* count as 2 (delete x_3, *whose Support* count is 1 and smaller than 2) and obtain the candidate item sets C1 given in Table 3.

(c) Scan the *and-term* sets once more and count the *support*s L2 of C1 presented in Table 4.

Table 4: *Support* count for variable sets L2.

Item	Support count
$\{x_1\ x_2\}$	1
$\{x_1\ x_4\}$	2
$\{x_1\ x_5\}$	1
$\{x_1\ x_6\}$	1
$\{x_2\ x_4\}$	3
$\{x_2\ x_5\}$	2
$\{x_2\ x_6\}$	2
$\{x_4\ x_5\}$	1
$\{x_4\ x_6\}$	1
$\{x_5\ x_6\}$	3

Table 5: *Support* count for variable sets C2.

Variable item	*Support* count
$\{x_1\ x_4\}$	2
$\{x_2\ x_4\}$	3
$\{x_2\ x_5\}$	2
$\{x_2\ x_6\}$	2
$\{x_5\ x_6\}$	3

Table 6: *Support* count for variable sets L3.

Variable item	*Support* count
$\{x_2\ x_5\ x_6\}$	2
$\{x_1\ x_2\ x_4\}*$	1
$\{x_1\ x_5\ x_6\}*$	1
$\{x_2\ x_4\ x_5\}*$	1
$\{x_2\ x_4\ x_6\}*$	1
$\{x_4\ x_5\ x_6\}*$	1

Table 7: *Support* count for variable sets C3.

Variable item	*Support* count
$\{x_2\ x_5\ x_6\}$	2

(d) Acquire the candidate item sets C2 according to the *min_sup* and delete the items whose *support* count is less than 2: $\{x_1, x_2\}$, $\{x_1, x_5\}$, $\{x_1, x_6\}$, $\{x_4, x_5\}$, $\{x_4, x_6\}$ as shown in Table 5.

(e) Scan the *and-term* sets again and compute the *support*s L3 of C2 as Table 6. According to the property that the superset of an infrequent item sets is not a frequent one, the items ($\{x_1, x_2\}$, $\{x_1, x_5\}$, $\{x_1, x_6\}$, $\{x_4, x_5\}$, and $\{x_4, x_6\}$) whose *support*s are less than *min_sup* can be deleted to form the final item sets C3 exhibited in Table 7.

To the nonempty subsets $\{x_2, x_5\}$, $\{x_2, x_6\}$, $\{x_5, x_6\}$, $\{x_2\}$, $\{x_5\}$, and $\{x_6\}$ of $\{x_2, x_5, x_6\}$, each confidence of the association rules of $\{x_2\ x_5\} \longrightarrow x_6$, $\{x_2\ x_6\} \longrightarrow x_5$, $\{x_5\ x_6\} \longrightarrow x_2$, $x_5 \longrightarrow \{x_2\ x_6\}$, and $x_6 \longrightarrow \{x_2\ x_5\}$ can be acquired, respectively, when the *min_conf* is set to 0.60, as shown in Table 8. On the basis of the *confidence* coefficient in Table 8, it can be seen that x_6 must appear when *and-term* includes x_2 and x_5 because the *confidence*($\{x_2, x_5\} \longrightarrow x_6$) is equal to 1, and x_2 will appear in the *and-term*s at a probability of 0.67 if x_5 and x_6 are contained because *confidence*($\{x_5\ x_6\} \longrightarrow x_2$) is equal to 0.67. The meaning of other rules can be obtained in a similar way.

Table 8: *Confidence* of the nonempty proper subsets of $\{x_2, x_5, x_6\}$.

Relationship among items	Confidence coefficient
$\{x_2\ x_5\} \longrightarrow x_6$	Confidence=2/2=1.00
$\{x_2\ x_6\} \longrightarrow x_5$	Confidence=2/2=1.00
$\{x_5\ x_6\} \longrightarrow x_2$	Confidence=2/3≈0.67
$x_2 \longrightarrow \{x_5\ x_6\}$	Confidence=2/4=0.50
$x_5 \longrightarrow \{x_2\ x_6\}$	Confidence=2/3≈0.67
$x_6 \longrightarrow \{x_2\ x_5\}$	Confidence=2/3≈0.67

Thus, the Apriori method is inherently a brief search process with a clear disadvantage [14]: Apriori must scan the database repeatedly, which increases the I/O load. Accordingly, the number of candidate frequent item sets is promptly augmented, and the computing time is prolonged if there are large numbers of items. The FP tree (FP-growth), which can avoid repeated database searches, was proposed by Han Jiawei in 2000 [15]. The algorithm can realize the compression of sample data where the FP tree forms an overlapping stem with the same items. However, it may not make a conspicuous advance in mining efficiency of the association rule due to visiting the data repeatedly when the stem bifurcation of FP-growth is very large.

As the classical association rule mining algorithms, Apriori and FP-growth are widely used by researchers. Swarm intelligence, with the features of distributed control and high self-organization, is also applied in these kinds of domains [16]. Significant improvements can be made in combining association rule mining with swarm intelligence.

2.2. PSO Algorithm. PSO [17] is a parallel heuristic random search algorithm, which converges to the global optimal solution with larger probability. It is proved by practice that it is suitable for optimization in dynamic and multiobjective optimization environment. Compared with traditional simulated annealing, ant colony algorithm, and genetic algorithm, it has faster computing speed and better global search ability. At the same time, because of the unique memory of PSO, it can dynamically maintain the extraction process of the nonlinear Boolean function association term, dynamically track the current search, and adjust its search strategy. Therefore, PSO can achieve faster convergence in the extraction of association term for nonlinear Boolean functions.

PSO, as a method for globally searching for an optimal solution, was proposed by Kennedy and Eberhart to simulate the foraging behavior of a flock of birds. Each bird is considered a particle of the search space and it can determine the next position according to its own "flying experience" and the shared information among the birds.

There are m particles in D-dimensional space. The position vector of particle i is $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$ and its velocity impacting the next motion state can be expressed as $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,D}\}$ in the feasible solution space. Every

particle has a fitness value decided by an objective function. The position X_i and the optimal position $p_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,D}\}$ with the best fitness value (also known as *pbest*) can both be acquainted with their "own flying experience". In addition, particles can achieve a global optimal position $p_g = \{p_{g1}, p_{g2}, \ldots, p_{gD}\}$ also called *gbest* from the shared information. The velocity and position of i in the jth dimension can be refreshed by (4), (5):

$$V_{i,j}^{t+1} = w \times V_{i,j}^{t} + c_1 \times r_1 \times \left(pbest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2 \times r_2 \times \left(gbest - x_{i,j}^{t}\right) \quad (4)$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + V_{i,j}^{t+1} \quad (5)$$

The constant inertia weight w equips the particle with an extended searching ability in the solution space. The capability for self-summary and learning of the excellent particles can be indicated by c_1 and c_2, whose values range within $(0, 4]$ and are usually equal to 2. r_1 and r_2 are the pseudorandom numbers at $(0, 1]$. The maximum velocity is called v_{max} and the particle can reach maximum velocity $v_{max,j}$ in the jth dimension. The schematic movement diagram of particle i in 3-dimensional space is shown in Figure 1.

The process for the PSO is introduced as the following.

(1) Initiation.

(a) Set the basic parameters: c_1, c_2, w, r_1, r_2, and v_{max}, particle scale p, maximum circulation k_{max}, and the accuracy requirement δ.

(b) Initialize the positions and velocities of the particles randomly.

(c) $k = 1$, $i = 1$.

(2) Global optimization.

(a) Calculate the fitness value f_i^k via x_i^k.

(b) If $f_i^k < fbest_i$, $fbest_i = f_i^k$ and $pbest_i = x_i^k$; otherwise $fbest_g = f_i^k$ and $pbest_g = x_i^k$.

(c) When $k > k_{max}$ or the accuracy meets the requirement, go to step (3).

(d) $i = i+1$ and if $i > p$, $k = k+1$, $i = 1$, then go to step ((2)(a)).

(3) Output the results and end the program.

3. Shared Variables Extraction Algorithm Based on PSO

According to the analysis in Sections 2.1 and 2.2, the Apriori algorithm searches for the optimal solution globally, while swarm intelligence is a local solution search process. The SVE-PSO method proposes applying their features in combination to improve the mining efficiency and is introduced in this section.

3.1. TSP of Shared Variables

Definition 5. Swapping refers to exchanging the positions of two items. For the TSP of m destinations, the solution sequence can be denoted as $S = \{a_1, a_2, \ldots, a_i, \ldots, a_j, \ldots, a_m\}$. If i and j are the positions of a_i and a_j in the sequence, the operation $S' = S \circ <i, j>$ is called *swapping*, which exchanges the positions of a_i and a_j but not the others. The operation $\circ$ is considered multiplication here. S' is the fresh solution sequence after swapping, and $<i, j>$ is the recon.

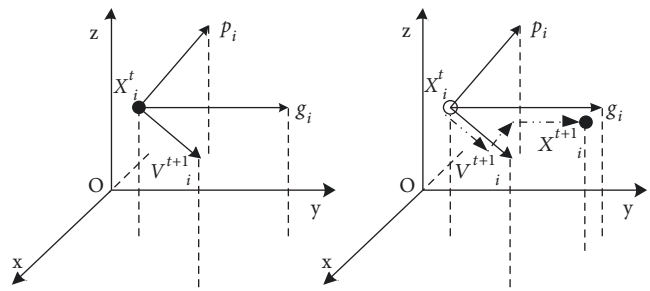


Figure 1: Movement diagram for particle i in three-dimensional coordinate.

Definition 6. Swapping sequence refers to an ordered queue consisting of one or more different *swapping*(s). This indicates that all recons act on the solution sequence one by one when the swapping sequence is operated on the solution sequence. Furthermore, the swapping sequences are equivalent to swapping sequences after swapping via different swapping sequences, where the same fresh solution is obtained. The set of all equivalent swapping sequences is the equivalent swapping set and the shortest swapping sequence contains the least recons.

The extraction of shared variables can be converted to the TSP. The variables in the database can be treated as the destination, and the undirected graph is composed of all the variables that can be mapped.

SVE-PSO is supposed to find the frequent item sets satisfying the maximum *support* resembling the shortest path of an undirected graph. The velocity and position of particle i can be updated by (6) and (7) for the introduction of recon and the swapping sequence.

$$V_{i,j}^{t+1} = V_{i,j}^{t} \circ \alpha\left(pbest_{i,j}^{t} - x_{i,j}^{t}\right) \circ \beta\left(gbest - x_{i,j}^{t}\right) \quad (6)$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} \circ V_{i,j}^{t+1} \quad (7)$$

α and β are random data in the interval $(0, 1)$. $\alpha(pbest_{i,j}^{t} - x_{i,j}^{t})$ means the swapping sequence $(pbest_{i,j}^{t} - x_{i,j}^{t})$ can be preserved at the probability of α. $pbest_{i,j}^{t}$ achieves a greater effect if α possesses a bigger value to achieve $\beta(gbest - x_{i,j}^{t})$.

Three steps are used to complete the extraction of shared variables in consideration of the Apriori algorithm and the PSO.

(1) Build the undirected graph. A 1-item frequency set must be included in all frequent item sets. All the frequent 1-item sets can be found by scanning the variable base and the infrequent 1-item sets should be deleted which can compress the valid nodes and lower the resource consumption. The number of valid nodes is 50 if the *support*s of half the variables are lower than *min_sup* in 100 variables $x_0 \sim x_{99}$.

```
Input: Term 1, Term 2, ..., TermN; n; k_max; fit; min_sup; V_max.
Begin
  Begin Initialization
    basic parameters setting
    and-term database encoding
    undirected graph stabling
  End Initialization
  Start:
    V_{i,j}^t = v, X_{i,j}^t = x
    Search the nodes of undirected graph
    If (V_{i,j}^{t+1} in the shortest_swapping_order)
      Swapping X_{i,j}^t with X_{i,j}^{t+1}
      Extract the associate terms from candidate set and calculate the value of fitness value
      Refresh pbest_{i,j}^t and gbest_{i,j}^t
    Else
      Refresh V_{i,j}^t and X_{i,j}^t
      If (optimal number of candidate set reaches the maximum)
        If (valid associate term)
          Output the term
        Else
          If (having searched all particles)
            Break the loop
          Else
            Refresh V_{i,j}^t and X_{i,j}^t
      Else
            Chang the association logic
End
Output: all association items in Boolean variables A_1, A_2, ..., A_t.
```

Algorithm 1: Pseudocode description of SVE-PSO algorithm.

(2) Generate the candidate item sets. The fresh solution $x_{i,j}^{t+1}$ can be gained in accordance with the current solution $x_{i,j}^{t}$ under the principle of maximum *support*.

(a) Assign an initial solution and swap the sequence for each particle.

(b) Search the nodes according to their *support*s.

(c) Renew the velocity and the position at time t+1.

(d) Decide on the added item $x_{i,j}^{t+1}$, as the potential item, based on whether it contains frequent item sets and velocity $V_{i,j}^{t+1}$ is the shortest swapping sequence. Otherwise, swap the positions through recon.

(3) Extract the association rules.

(a) Extract the association rules for all potential items in the candidate item sets and calculate the fitness value, also known as the *confidence*.

(b) Update the individual extreme value of the candidate items and the optimal value of candidate item sets.

(c) Output the complete and valid association rules; otherwise, update the optimal value and search again.

The flow diagram for the SVE-PSO algorithm is shown in Figure 2, and the pseudocode description of SVE-PSO algorithm is shown in Algorithm 1.

3.2. The Analysis of SVE-PSO. An example is used to analyze the SVE-PSO procedure. One specific NLBF includes N *and-term*s, which can be presented as $\{x_1, x_2\}$, $\{x_4\}$, $\{x_2, x_3, x_5\}$, $\{x_1, x_4, x_6\}$, ..., $\{x_6\}$, $\{x_2, x_3, x_7\}$. The process of SVE-PSO can be described as the following:

(a) Scan the variable database, delete the infrequent 1-item sets, and acquire the variables x_1, x_2, x_3, x_4, x_5, and x_6 whose *support*s are greater than *min_sup*. The initial undirected graph can be seen in Figure 3 (1).

(b) Choose three particles from the particle swarm randomly, and sort the item sets according to their *support*s. Assume the ranked sequence is $\{x_2, x_4, x_6, x_3, x_5, x_1\}$, $\{x_1, x_5, x_6, x_2, x_4, x_3\}$, and $\{x_3, x_6, x_5, x_4, x_1, x_2\}$ and that the initial paths of the three particles are $(x_2 \longrightarrow x_4 \longrightarrow x_6 \longrightarrow x_3 \longrightarrow x_5 \longrightarrow x_1)$, $(x_1 \longrightarrow x_5 \longrightarrow x_6 \longrightarrow x_2 \longrightarrow x_4 \longrightarrow x_3)$, and $(x_3 \longrightarrow x_6 \longrightarrow x_5 \longrightarrow x_4 \longrightarrow x_1 \longrightarrow x_2)$.

(c) When the initial position of particle 1 is x_2, determine whether the item $\{x_2\ x_4\}$ includes the frequent item sets or not, shown as Figure 3(2)(a). If the item does not include frequent item sets, swap item x_4 with x_6. Postulating that there are not any frequent item sets in $\{x_2, x_4\}$, as shown in Figure 3(2)(b), the variable x_4 will not be added to the final path, and the position of particle 1 is changed according to x_6.

(d) Judge whether the item $\{x_2\ x_6\}$ obtains frequent item sets or not as shown in Figure 3(2)(c) when the position is x_6. By carrying out the swapping and if there is no frequent item

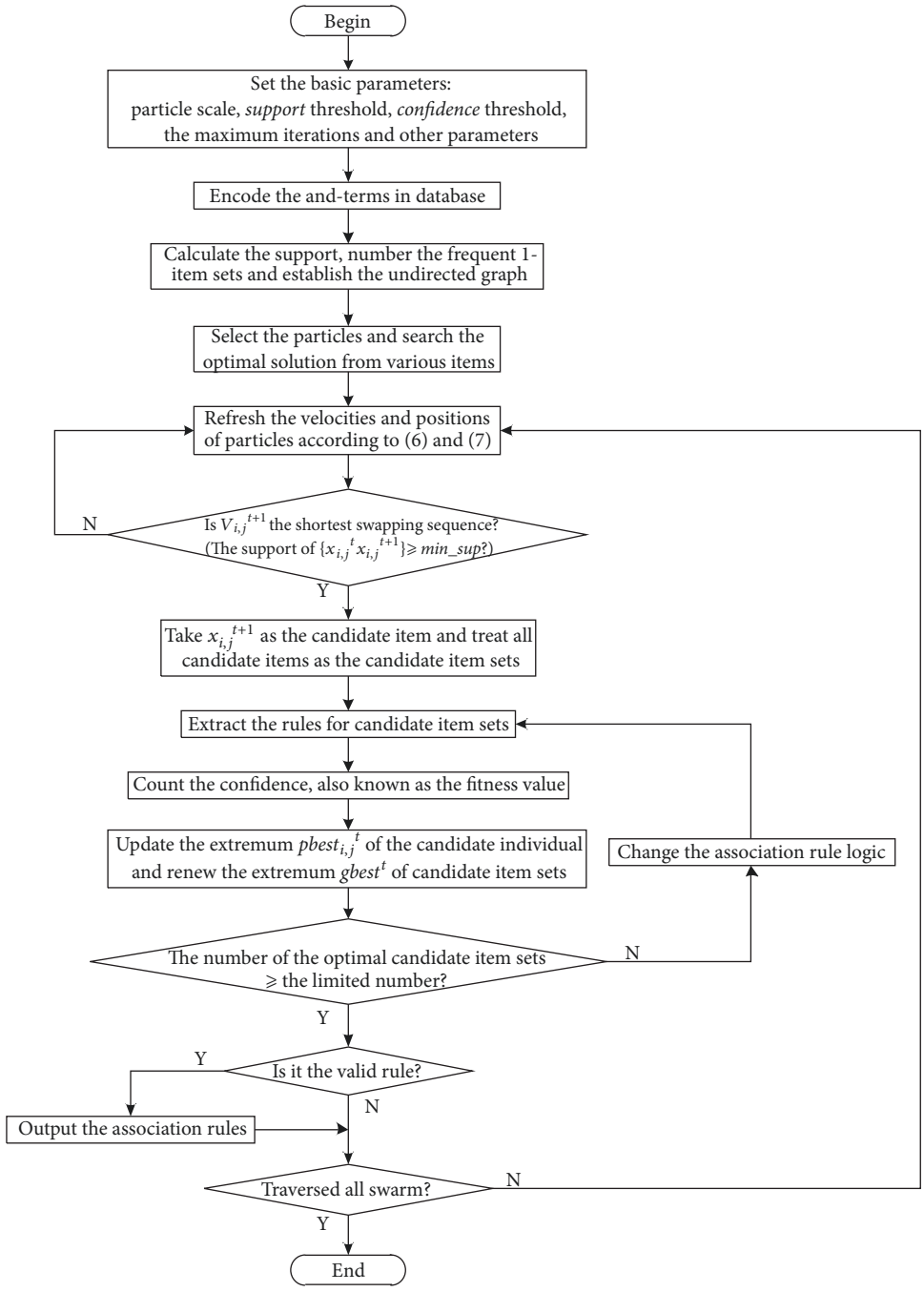


Figure 2: The procedure for SVE-PSO.

set, then delete variable x_6. If the item $\{x_2\ x_6\}$ is the frequent item, the variable x_6 can be reckoned in the final path and output $\{x_2\ x_6\}$ as the candidate frequent item sets shown in Figure 3(2)(d). The recon is <3, 2>.

(e) Determine whether the item $\{x_2\ x_6\ x_3\}$ contains frequent item sets or not as shown in Figure 3(2)(e). Suppose that the item does not include the frequent item sets; then perform the swapping operation and delete variable x_3. Here, the item $\{x_2\ x_6\ x_3\}$ is the frequent item, so x_3 is joined in the final path and output the item $\{x_2\ x_6\ x_3\}$ as the candidate item sets given in Figure 3(2)(f). The recon is <4, 3>.

(f) Decide whether the item $\{x_2\ x_6\ x_3\ x_5\}$ includes frequent item sets or not as shown in Figure 3(2)(g). If there are no frequent item sets in $\{x_2\ x_6\ x_3\ x_5\}$, the variable x_5 is deleted, as presented in Figure 3(2)(h).

(g) To confirm whether the item $\{x_2\ x_6\ x_3\ x_1\}$ possesses frequent item sets or not, as shown in Figure 3(2)(i), if there are no frequent item sets, abandon variable x_1 and carry out the swapping operation as shown in Figure 3(2)(j).

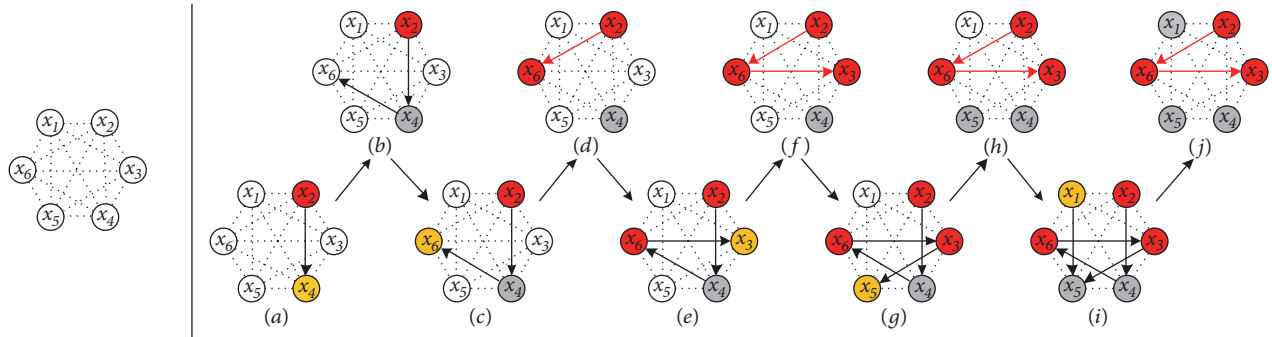


Figure 3: Process of SVE-PSO.

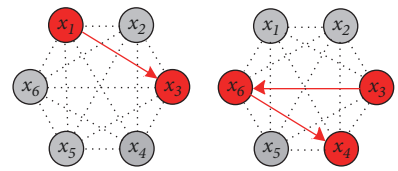


Figure 4: Final paths of particle 2 and particle 3.

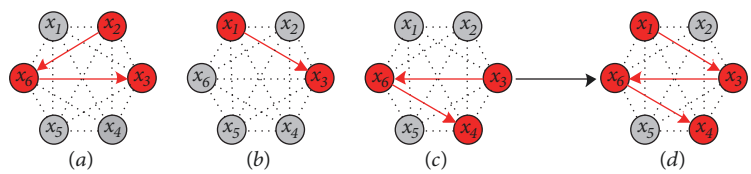


Figure 5: Cross of the final paths.

(h) Finally, all candidate item sets achieved via particle 1 are $\{x_2\ x_6\ x_3\}$, $\{x_2\ x_6\ x_3\}$, $\{x_2\ x_6\}$, $\{x_6\ x_3\}$, and $\{x_2\ x_3\}$. Its final path is $x_2 \longrightarrow x_6 \longrightarrow x_3$, and the swapping sequence can be expressed as <3, 2> ∘ <4, 3>.

(i) Similarly, the candidate item sets of particle 2 and particle 3 can also be gained: $\{x_1\ x_3\}$ and $\{x_3\ x_6\ x_4\}$, $\{x_3\ x_6\}$, $\{x_3\ x_4\}$. The final paths are $x_1 \longrightarrow x_3$, shown in Figure 4(a), and $x_3 \longrightarrow x_6 \longrightarrow x_4$, given in Figure 4(b).

(j) Calculate the fitness value of the candidate item sets and renew the individual historical best value and global optimization via the "flying experience". The final paths of particles 1, 2, and 3 are shown as Figures 5(a), 5(b), and 5(c), respectively. According to Figures 5(b) and 5(c), the items$\{x_1\ x_3\ x_6\}$ and $\{x_1\ x_3\ x_6\ x_4\}$ can also be taken as candidate item sets, as shown in Figure 5(d). Other candidates can be achieved in a similar way.

(k) Jump to step (a) to search other potential item sets, if all positions have not been searched completely by the particle swarm.

In this way, shared variables can be gained.

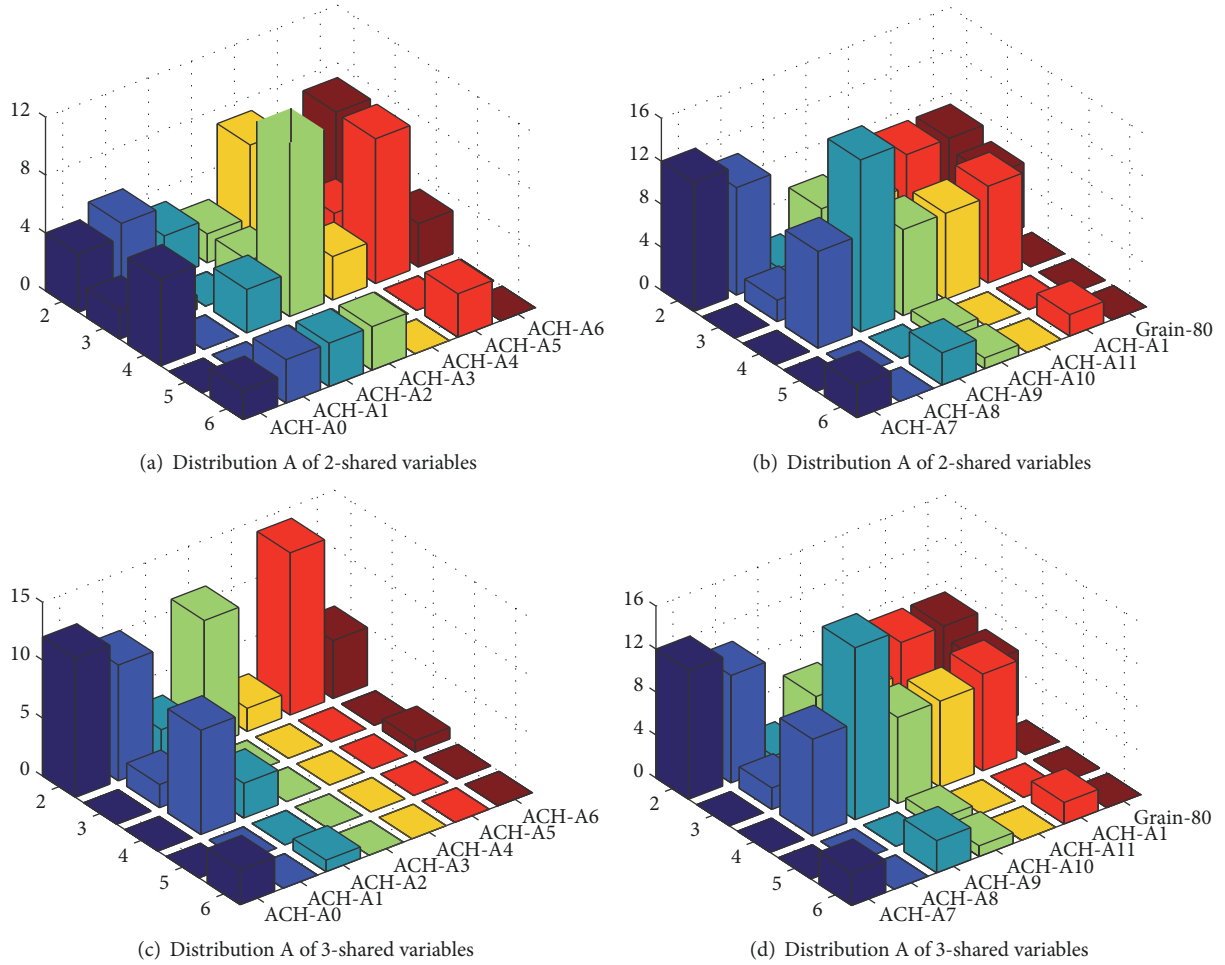
The direct purpose of the SVE-PSO method is to extract shared variables in various NLBFs, but extracting shared variables is not the final aim. The final aim is to design the hardware architecture of the NLBFs according to the shared variables that were extracted.

4. Hardware Implementation for NLBF

To complete the hardware architecture of NLBFs, the distribution of shared variables must be achieved. According to the distribution results and distribution laws, two kinds of logic elements are proposed.

Hardware implementation of the NLBFs is based on the distribution of shared variables. There are two basic parameters to measure the distribution: on the one hand, the frequency of the variables and the expression number for each certain shared variable in NLBFs should be obtained because not all shared variables are mapped as the shared inputs; on the other hand, the variety of shared variables and their frequency must also be determined to know the features of different NLBFs. All instances are given in Figure 6 as a distribution of 2-shared variables (Figures 6(a) and 6(b)) and a distribution of 3-shared variables (Figures 6(c) and 6(d)). In Figure 6, the X-axis denotes the various cryptographic algorithms using NLBFs, the Y-axis denotes the frequency of the shared variables, and the Z-axis denotes the variety of variables of specific shared frequency for Boolean expressions in corresponding cryptographic algorithms.

According to Figure 6, there are usually 2-shared and 3-shared variables, some situations may have 4-shared variables for a certain NLBF. In addition, the frequency of shared



(a) Distribution A of 2-shared variables

(b) Distribution A of 2-shared variables

(c) Distribution A of 3-shared variables

(d) Distribution A of 3-shared variables

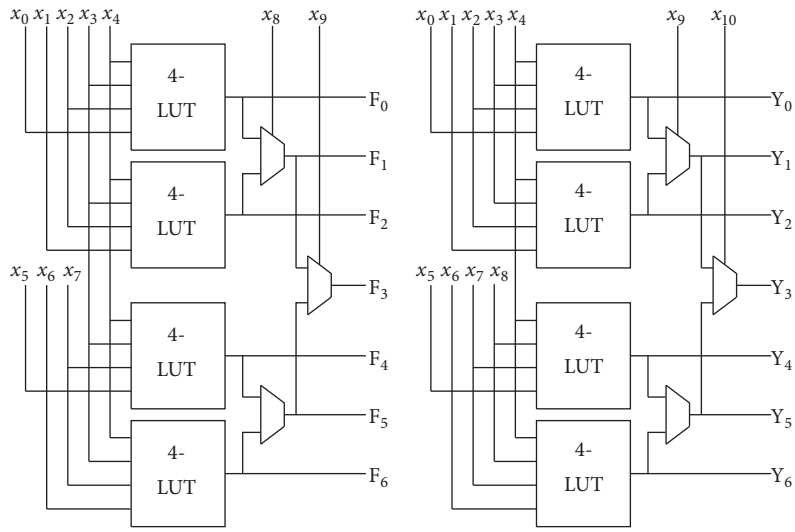
Figure 6: Distribution of the shared variables for NLBFs.

variables covers a range from 2 to 6. For example, ACH-A0 possesses 2~3 shared variables and the frequencies are 2, 3, 4, and 6 for 2-shared variables. The variety of 2-shared variables is four, two, six, and two for the frequencies 2, 3, 4, and 6. Every 2-shared variable does not appear five times in all NLBFs expressions of the ACH-A0. The characteristics of the 3-shared variables and 4-shared variables can also be obtained like the 2-shared variables.

There is some discrepancy among various NLBFs, due to the diverse principles, but they are also equipped with several similarities. First, the variable number they share does not exceed 4 and only ACH-A7 has 4-shared variables. Furthermore, the varieties of 2-shared variables change during 10 ~ 14 or 20 ~ 22, and the range of 3-shared variables is 6~7 or 11~16 after calculations. Finally, the variety of 2-shared variables whose frequency is four achieves the maximum value; thus, there are 3-shared variables in the two items, and there are 2-shared variables among four *and-terms* and 3-shared variables between *and-terms*.

According to the distribution of shared variables in various NLBFs, two cascade LUT-based logic elements named Element A and Element B are designed, as shown in Figures 7(a) and 7(b), respectively. The two logic elements of both include four 4-LUTs and possess the same cascading pattern, but they are different in terms of the number of input ports. Element A, shown in Figure 7(a), contains 10 input ports and 6 output ports; its four 4-LUTs include two shared input ports and two 4-LUTs in pairs with three shared input ports. Element B shown in Figure 7(b) increases 1 input port relative to Element A; it includes 11 input ports, and its four 4-LUTs include 1 shared input port and two 4-LUTs (in pairs) contain three shared input ports. As seen in the logic elements, there are two-level output structures that consist of three MUX2_1s.

At the same time, these two logic elements are comprised of the same hardware resource. The two logic elements need 64-bit configuration information and 63 MUX2_1s. The hardware resource is equal to 6-LUT but both logic elements are comprised of the input ports which can finish more types of Boolean functions than 6-LUT. For example, the number of variables in a Boolean function is more than 6 but less than 12, that is, the number of variables is from 7 to 11. If the number of variables is larger than 6 and less than 12, the Boolean function will need two 6-LUTs. However, if the number of variables does not exceed 10, it can be applied into the structure of Element A, and if the number of variables



(a) Element A with 10 inputs (b) Element B with 11 inputs

Figure 7: Logic elements of cascaded LUT-based logic elements.

Table 9: The distribution of item sets in NLBFs.

Item	1-term	2-term	3-term	4-term	5-term	6-term	17-term	63-term
Number	261	359	192	251	47	5	1	1

does not exceed 11, the Element B, which includes 11 input ports, can be used.

5. Experimental Results

Several experiments are carried out to evaluate the performance of SVE-PSO from various aspects, and the characteristics of the shared variables are achieved according to the association rules mined by SVE-PSO.

5.1. Performance Test for the SVE-PSO Method. To estimate the performance of SVE-PSO, a test platform with an 8G memory, 1Tb hard disk, and Windows 7 operating system must be built. In addition, the tested term sets are gained by doing the statistical work on the *and-terms* for various NLBFs after analyzing multiform cryptographic algorithms such as ACH series algorithms, LILI, Grain, and Trivium. The tested term sets can be seen in Table 9, where the 3-term indicates the third-order *and-term*s with three variables. The corresponding 192 value means the amount of three-order *and-term*s in various NLBFs of multiform cryptographic algorithms, etc. Then, the assessments are obtained by comparison with the classical mining algorithms.

There are four parameters involved in the association rule mining: the association rule number, particle swarm scale, *support*, and *confidence*. This paper tested the relationship of association rule number with *support*, *confidence*, and particle swarm scales.

(a) The relationship of the association rule number with *support*: the particle swarm scale is 40, the maximum iteration is 200, and the *min_conf* is 0.30. The results are shown in Figure 8(a).

(b) The relevance between the association rule number and *confidence*: the particle swarm scale is 40, the iteration is 200, and the *min_sup* is 0.30. The change rule is shown in Figure 8(b).

(c) The relationship of the association rule number with particle swarm scale: the iteration is 200 and the *min_conf* and *min_sup* values are both 0.30. The changes are shown in Figure 8(c).

As shown in Figure 8(a), the number of association rules mined by SVE-PSO increases with the decrease in the *support* function under a constant particle swarm scale, iteration times, and *confidence*. In addition, because the Apriori method is adapted as a global search method, the rules that are achieved are marginally more than SVE-PSO. From Figure 8(b), when the swarm possesses an unchanged particle swarm scale, unchanged iteration times, and *support*, the association rule number is also added by the decline in *confidence*. Meanwhile, the ratio of the association rule number can approach 80% when the *confidence* varies in the range 0.20~0.60. Figure 8(c) shows that the number of association rules augmented by a particle swarm scale can even attain the same value compared with Apriori or FP-growth under fixed iteration times, *support*, and *confidence*.

At the same time, the performance of SVE-PSO is also compared with the Apriori or FP-growth shown in Figure 9. The relationship between the operation time and iteration time is investigated via a steadfast particle swarm scale (40), *min_sup* (0.30), and *min_conf* (0.30). The operation time is prolonged, and the number of association rules is increased significantly. The performance achieves a significant lead over Apriori and FP-growth. For example, the ratio of the rules is

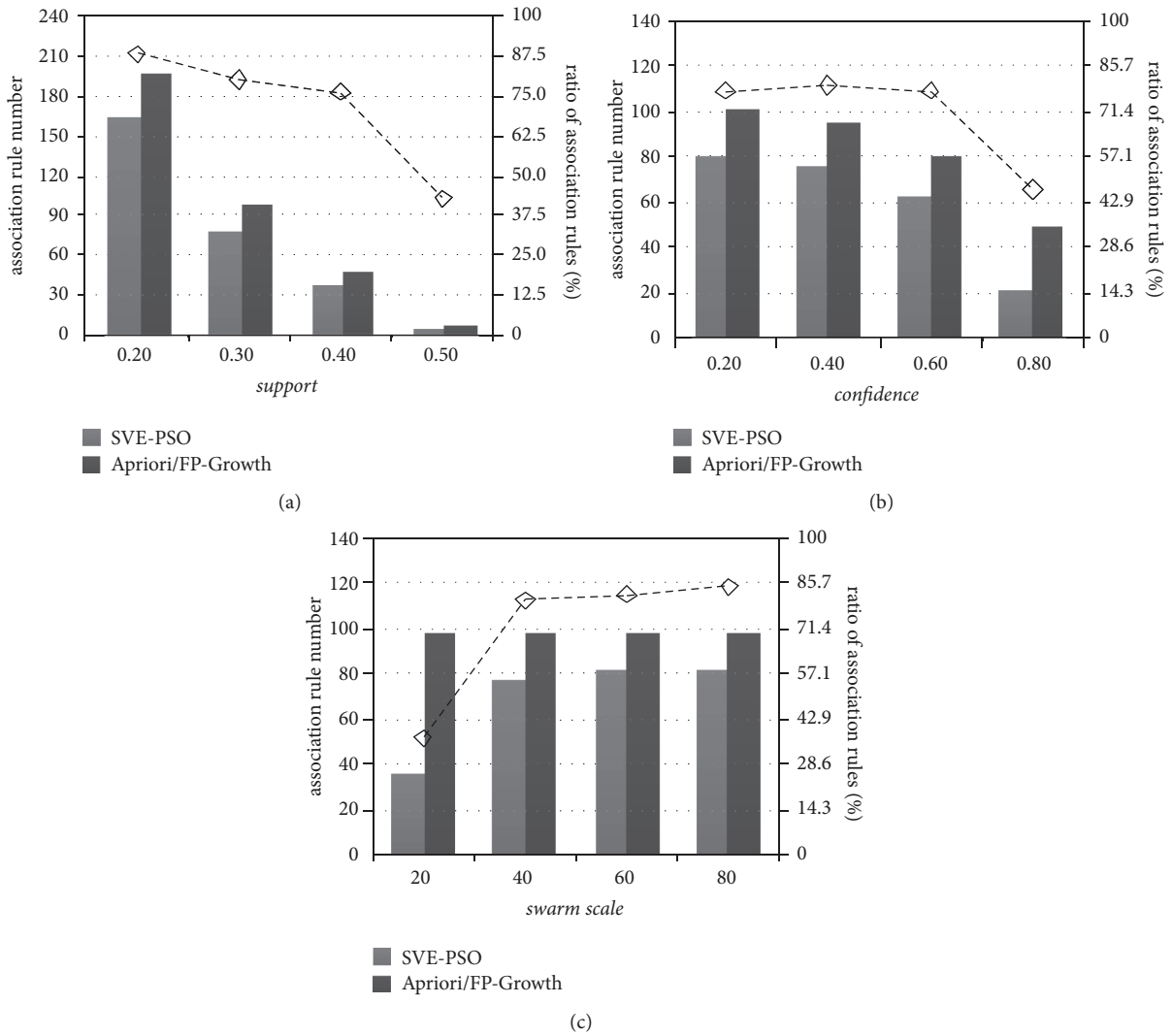


Figure 8: Relationship of the association rule number with *support*, *confid*ence, and particle swarm scale. (*a*) Relationship of the association rule number with *support*. (*b*) Relationship of the association rule number with *confidence*. (*c*) Relationship of the association rule number with the particle swarm scale.

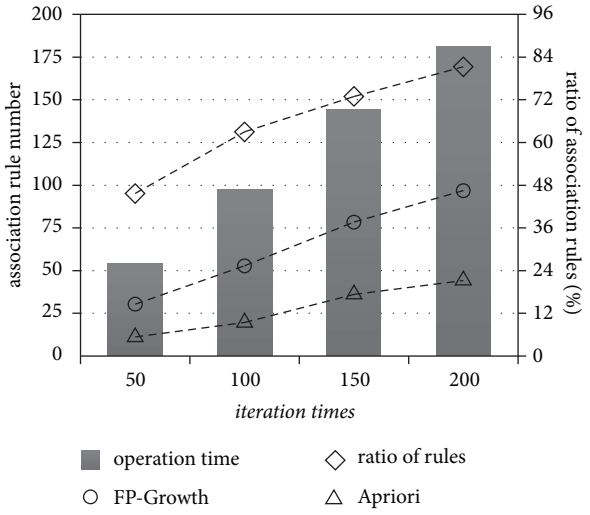


Figure 9: Relationship between operation time and iteration time.

80.41% but the operation time is only 21.47%, compared to Apriori with 200 iterations.

According to Figures 8 and 9, it can be concluded that the proposed SVE-PSO method can achieve a significantly higher efficiency and better results with rational parameters than traditional methods.

5.2. Performance Test for Hardware Elements. To evaluate the performance of the designed hardware elements, the logic elements numbers of NLBFs used are gained under the parallelism of 1, as is shown in Figure 10. It can be seen that the numbers of logic elements are various in different NLBFs but Elements A and B consume the same number of logic elements in different NLBFs. A majority of the NLBFs require only 3~4 logic elements but ACH-F limits the logic elements to at most 12.

$$\%UtilizedLUTs = \frac{\#used\ LUT_4}{\#all\ LUT_4} * 100 \tag{8}$$

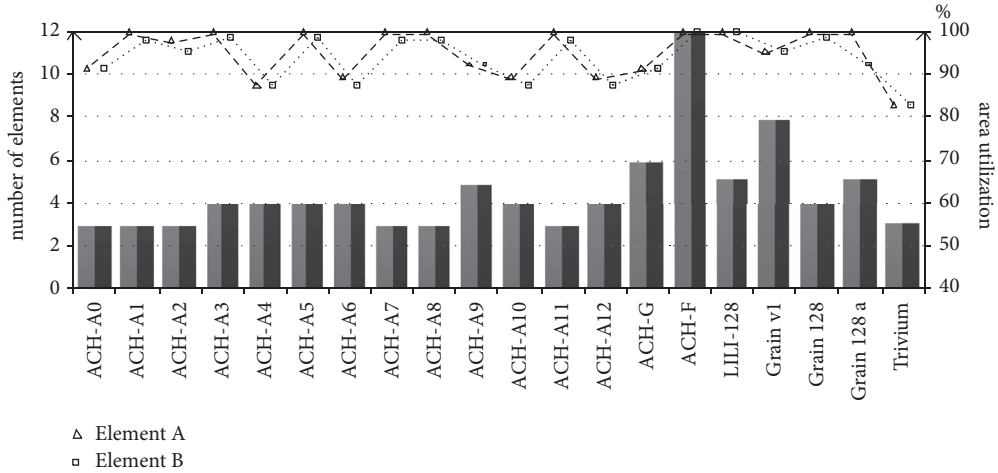


Figure 10: Number of logic elements in various NLBFs.

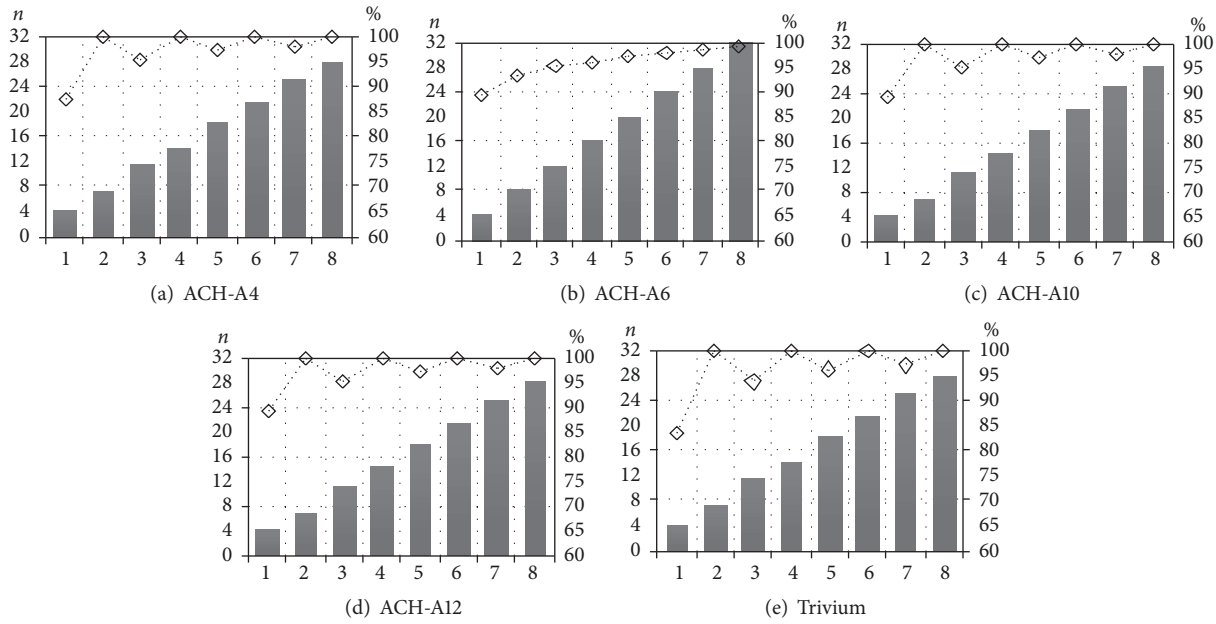


Figure 11: The number of logic elements and area utilization for different NLBFs.

At the same time, the area utilizations of Element A and Element B are measured in (8) and characterized as dotted lines in Figure 10. The area utilization of several NLBFs can reach 100% in both Element A and Element B. The area utilization of almost all NLBFs can achieve 90% except for ACH-A4, ACH-A6, ACH-A10, ACH-A12, and Trivium. The corresponding area utilizations for Element A and Element B are 87.5%, 89.1%, 89.1%, 89.1%, and 83.3% and 87.5%, 87.5%, 87.5%, 87.5%, and 83.3%. The area utilization of Element A is not less than Element B following the same trend.

In addition, because various NLBFs possess different parallelisms, the number and area utilization of logic element in Element A and Element B, under different parallelisms, should be determined. This work determines the logic element number and area utilization of ACH-A4, ACH-A6, ACH-A10, ACH-A12, and Trivium for different parallelisms, respectively, as shown in Figure 11. The horizontal axis in Figure 11 indicates the parallelism of NLBFs, the histogram represents the number of logic elements in NLBFs, and the graph stands for the area utilization of NLBFs via different parallelisms. Meanwhile, there are two principles that should be considered. The first is that the area utilization increases with the parallelism, because some unapplied 4-LUTs in logic elements would be used when the parallelism increases. The second is that the area utilization usually reaches 100% when the parallelism is even, but the ratio usually cannot achieve 100% when the parallelism is odd. The reason for this is that when the parallelism is odd, there are two 4-LUTs in logic element that cannot be used. When the parallelism is an even number, the two unused 4-LUTs will be used, and the ratio can reach 100%.

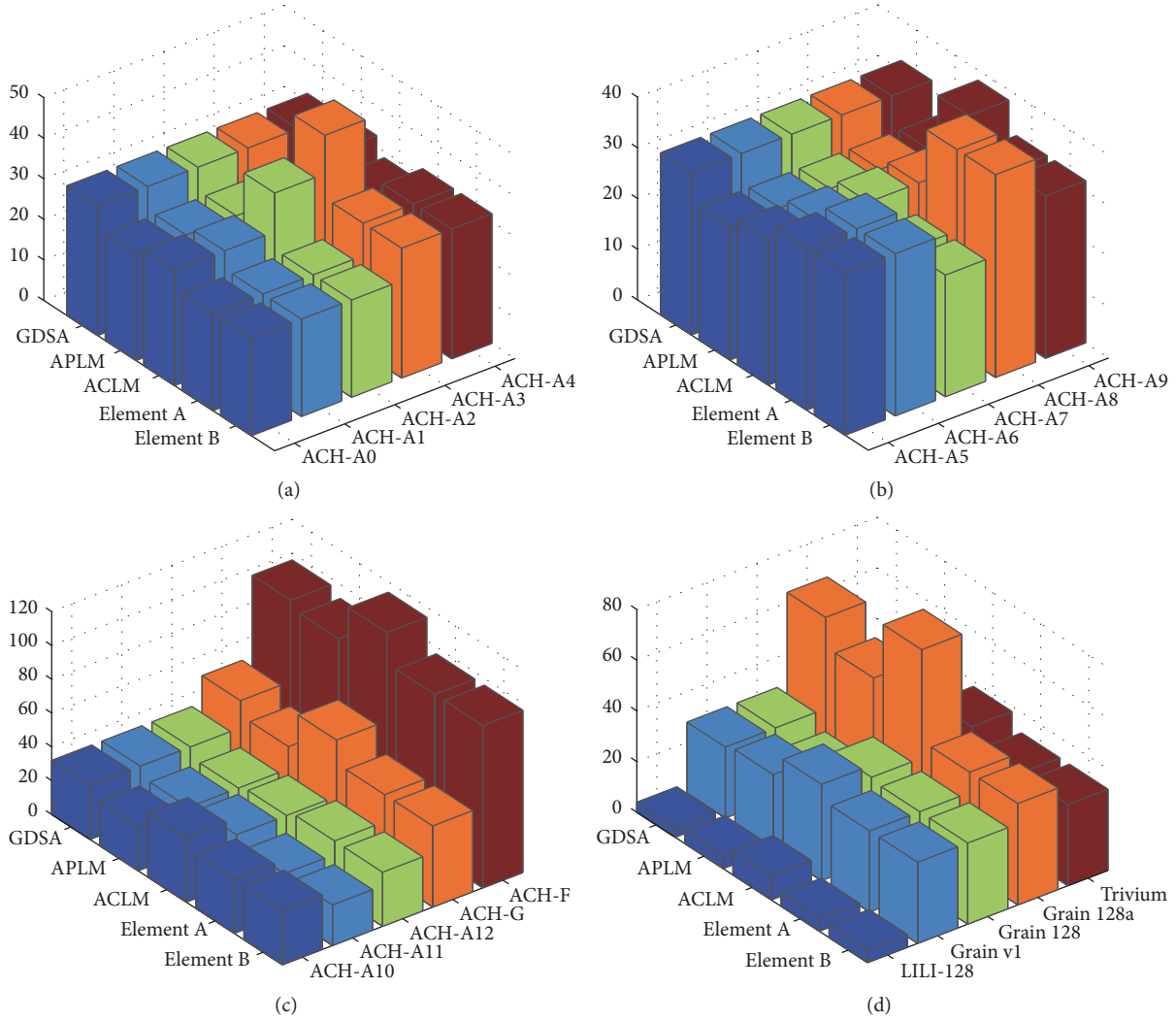


Figure 12: Comparison results of logic element number in different NLBFs.

Finally, to inspect the performance of Element A and Element B, the numbers of logic elements used in different NLBFs are compared to GDSA [18], APLM [19], and ACLM [20], as shown in Figure 12. The GDSA design is based on one 6-variable LUT-structure. The APLM design is based on three 4-LUTs structure with 5 unaided inputs, 4 shared inputs, and 3 outputs. The ACLM design is based on four 4-LUTs structure with 5 unaided inputs, 4 shared inputs, and 5 outputs. From Figure 12, most results of Element A and Element B are better than the GDSA and ACLM, because Element A and Element B are designed according to the more befitting distribution characteristics of the shared variables in various NLBFs. They also have more inputs and outputs to sustain more NLBFs processes in parallel. Some results of Element A and Element B are worse than APLM. This is because the tests are simply aimed at indie arithmetic. Though APLM possesses the lowest number of computing units, the applicability and flexibility are much worse than Element A and Element B.

6. Conclusion

This paper develops an algorithm of SVE-PSO that combines the concepts (e.g., frequent item set, confidence, etc.) of the Apriori method and "swarm intelligence" from biology to further improve the algorithm efficiency. Furthermore, the distribution of shared variables is assembled for various NLBFs, and according to it two cascade LUT-structure hardware elements are projected to satisfy the computation of NLBFs. Finally, experiments are conducted to verify the effectiveness of the SVE-PSO method, and the area utilizations of Element A and Element B expended by NLBFs are measured. To summarize, the integration of the SVE-PSO method and the two cascade LUT-structure hardware circuits has a better performance than other methods and can be used widely in cryptogrammic processors to enhance NLBF processing and mapping performance.

The algorithm of SVE-PSO developed is based on resource priority in NLBF adapters design. So in the future, the

proposal will be improved not only to optimize resource occupancy, but also to optimize other performance aspects, such as optimization of adaptation efficiency, and the execution efficiency of the runtime of the structure. Furthermore, many intelligent algorithms (e.g., genetic algorithm, neural network algorithm, etc.) also will be studied intensively to effectively solve the problem of multiobjective optimization. It will play a positive role in hardware structure and adapter optimization.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] M. Hutton, J. Schleicher, D. Lewis et al., "Improving FPGA Performance and Area Using an Adaptive Logic Module," in *Field Programmable Logic and Application*, vol. 3203 of *Lecture Notes in Computer Science*, pp. 135–144, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[2] J. H. Anderson and Q. Wang, "Area-efficient FPGA logic elements: Architecture and synthesis," in *Proceedings of the 2011 16th Asia and South Pacific Design Automation Conference, ASP-DAC 2011*, pp. 369–375, Japan, January 2011.

[3] P. Kitsos, N. Sklavos, K. Papadomanolakis, and O. Koufopavlou, "Hardware implementation of bluetooth security," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 21–29, 2003.

[4] G. Gong, M. D. Aagaard, and X. Fan, *Ligheweight stream cipher cryptosystems: US*, 2015.

[5] A. M. Daniel, J. Petro, and T. M. Millhollon, *High speed cryptographic combining system, and method for programmable logic devices*, 2015.

[6] V. Betz and J. Rose, "How much logic should go in an FPGA logic block?" *IEEE Design & Test of Computers*, vol. 15, no. 1, pp. 10–15, 1998.

[7] E. Ahmed and J. Rose, "Effect of LUT and cluster size on deep-submicron FPGA performance and density," in *Proceedings of the FPGA 2000: ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 3–12, February 2000.

[8] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, 2004.

[9] SD. Pable, A. Lmran, M. Hansan, and A. Lslam, "Performance optimization of LUT of subthreshold FPGA in deep submicron[C]," in *Proceedings of the IEEE International Conference on Computer Communication Technology*, pp. 64–69, 2010.

[10] T. N. Kumar, H. A. Almurib, and F. Lombardi, "A novel design of a memristor-based look-up table (LUT) for FPGA," in *Proceedings of the 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 703–706, Ishigaki, Japan, November 2014.

[11] W. Zhouchuang, D. Zibin, L. Wei, and C. Xin, "Reconfigurable design for NBF based on optimal area utilization model," in *Proceedings of the 10th IEEE International Conference on Anti-Counterfeiting, Security, and Identification, ASID 2016*, pp. 74–78, China, September 2016.

[12] Z. Dai, Z. Wang, W. Li, L. Nan, and H. Xiong, "Hardware implementation and utilization model research for reconfigurable non-linear boolean function[J]," *Journal of Electronics Information*, 2016.

[13] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases[C]," in *Proceedings of the The 20th International Conference on Very Large Data Bases, Santiago de Chile*, pp. 487–499, Chile, 1994.

[14] Y. Wang, *Information communication network alarm association rule mining algorithm based on swarm intelligence*, Taiyuan University of Technology, 2015.

[15] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *SIGMOD Record*, vol. 29, no. 2, pp. 1–12, 2000.

[16] Yang Wang, Guocai Li, Yakun Xu, and Jie Hu, "An Algorithm for Mining of Association Rules for the Information Communication Network Alarms Based on Swarm Intelligence," *Mathematical Problems in Engineering*, vol. 2014, Article ID 894205, 14 pages, 2014.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Western Australia, November-December 1995.

[18] W. Zhouchuang and D. Zibin, "Global directional search algorithm adapting NLBF sequence cryptogram efficiently," *Journal of Computer Application*, 2016, ISSN 1001.9081.

[19] J. Xiangjun, X. Chen, and D. zibin, "Design and realization of an improved hardware with non-liner Boolean function," in *Computer Applications and Sofhare*, vol. 31, 7 edition, 2014.

[20] W. Zhouchuang and D. Zibin, "Reconfigurable Design for NBF Based on Optimal Area Utilization Model," in *The International Conference on ASID*, 2016.